Optimal adaption for Apache Cassandra

Emiliano Casalicchio, Lars Lundberg, Sogand Shirinbab
Department of Computer Science and Engineering
Blekinge Institute ot Technology
Karlskrona, Sweden
Email: emc@bth.se, llu@bth.se, sogand.shirinbab@bth.se

***Abstract*—Apache Cassandra is a NoSql database offering high scalability and availability. Among with its competitors, e.g. Hbase, SympleDB and BigTable, Cassandra is a widely used platform for big data systems. Tuning the performance of those systems is a complex task and there is a growing demand for autonomic management solutions. In this paper we present an energy-aware adaptation model built from a real case based on Apache Cassandra and on real application data from Ericsson AB Sweden. Along with the optimal adaptation model we propose a sub-optimal adaptation algorithm to avoid system perturbations due to re-configuration actions triggered by subscription of new tenants and/or the increase in the volume of queries. Results shows that the penalty for an adaptation mechanism that does not hurt the system stability is between 20 and 30% with respect the optimal adaptation.**

***Keywords*—Autonomic computing; self-adaptation; energy-awarness; optimization; Apache Cassandra; Big Data**

I. Introduction

The resource management for Apache Cassandra base platforms is a challenging task and the complexity increase when multitenancy is considered. In this paper we focus on a provider of a managed Apache Cassandra service offered to support Big Data enterprise applications. Applications using the service submit NoSql queries (operations in what follow) at a specific rate. Each application requires a minimum throughput and a certain level of data replication to cope with nodes failures. To satisfy these customer's requirements the service provider has to properly plan the capacity and the configuration of each Cassandra virtual datacenter. On the other side, the service provider want to minimize its power consumption. Therefore, it should find the optimal placement of the Cassandra vnodes, to use as few physical machines (PM) as possible. To solve this problem we propose a model that orchestrate horizontal scaling (e.g add/remove Cassandra vnodes), vertical scaling (adding computing power (e.g. virtual cpu/memory) and optimal placement of vnodes on the cloud infrastructure. The model is designed to be embedded in the planning phase of a MAPE-K controller and it bases the adaptation decisions on two parameters that are easy to be collected and that are: the vnodes throughput and the CPU usage. In the last year has been proposed many research works on measuring (e.g. [1] and [2]) and managing the performance NoSql distributed data stores such as Cassandra. Many studies focus on the horizontal scalability feature offered by such databases, e.g. [3]–[8]. Few studies consider vertical scaling, e.g. [5], [7] and configuration tuning [7]–[10]. While Horizontal scaling, vertical scaling and configutation tuning approaches are somentime mixed, optimal placement is never considered in combination with the other adaptation strategies. However in literature there are many research work on the optimal placement of VMs on PMs. e.g. [11]–[15] With respect to the literature, this paper introduces three main novelties: We consider the adaptation of a multi tenants Cassandra-based system. We propose an energy-aware run time adaptation model that orchestrates horizontal scaling, vertical scaling and optimal placement of vnodes on the cloud infrastructure. The proposed model is built considering the best practice for the deployment and mangement of Apache Cassandra virtual data centers. We parameterize the proposed model using real data collected from a testbed running Ericsson AB Sweden's specific workload. Experiments shown: the effectiveness of the adaptation model; its limitations in term of unwanted system reconfiguration actions that could degrade the system performances; the tradeoff between having zero-unwanted-reconfiguration and globally minimizing the power consumption.

The paper is organized as in what follow. Next section II introduces the optimal adaptation model. The heuristic to find a sub-optimal solution for the problem is presented in Section III. Performance metrics and the experimental results are presented in Section IV. Finally, Section V provides concluding remarks.

II. Adaptation model

In this section we present the adaptation model formulated as an optimization problem. In this respect, we need to define models for: the workload and SLA; the architecture; the throughput and the utility function. Finally, we define the optimization problem. The solution of the optimization problem provides the optimal (or suboptimal) adaptation policy that, for each tenant, specify:

- the size of the Cassandra virtual datacenter in terms of number of vnodes
- the configuration of vnodes, e.g. in terms of CPU capacity
- the placement of vnodes on the physical nodes.

The periodic, or event based evaluation of the optimisation problem provides a runtime adaptation policy for the Cassandra service provider.

A. Workload and SLA Model

The system workload consist of a set read (R), write (W) and read & write (RW) operation requests. Such operation requests are generated by the N independent applications and we assume that each application i generate only a type $l_i \in \mathcal{L} = \{R, W, RW\}$ of requests. If $l_i = R$ or $l_i = W$

we have 100% R or W requests. In case $l_i = RW$ the workload is composed of 75% R and 25%W requests. In literature have been considered more sophisticated workloads that include specific case of read and write requests such as scan and update requests. We limit the study to the set $\mathcal{L}$ above defined because the model we propose can deal with any type of operation requests. Requests of type l_i are generated at a given rate and therefore each application need that a throughput T_i^{min} (operations per second) is guaranteed by the provider. Moreover, each application need a specific level of data redundancy specified by the data replications factor D_i (`replication_factor` is a configuration Cassandra parameter). Summarizing, the SLA contractualized between the tenant and the service provider can be modelled by the tuple $\langle l_i, T_i^{min}, D_i \rangle$.

B. Architecture model

We consider a datacenter consisting of H homogeneous physical machines (PMs) installed at the same geographical location. We assume each PM h has a nominal CPU capacity C_h measured in number of available cores. We assume that our scenario is not memory bound and therefore we do not model the memory capacity. Each Cassandra vnode run on a VM of type j configured with c_j virtual cores. We consider a set of V VM configurations. For example, in Table I are reported three different configuration for a VM ($V = 3$). A change from configuration j_1 to j_2 is modelled as the replacement of the VM of type j_1 with a VM of type j_2. However, in a real setting, hypervisors such as VMWare allow to change at runtime the number of cores associated to a VM without the need to shutdown the VM. We do not consider the case of over-allocation, that is the maximum number of virtual cores allocated on PM h is equal to C_h. As suggested by the Cassandra management best practice we assume that a Cassandra virtual datacenter is composed of n_i homogeneous Cassandra virtual nodes where $n_i \geq D_i$ and at least D_i out of n_i vnodes must run on different physical machines. For each application i these three constraints are modelled by the following equations:

$$\sum_{j \in \mathcal{J}} y_{i,j} = 1, \quad \sum_{j \in \mathcal{J}, h \in \mathcal{H}} x_{i,j,h} \geq D_i \text{ and } \sum_{h \in \mathcal{H}} s_{i,h} \geq D_i$$

where: $y_{i,j}$ is equal to 1 if application i use a VM configuration j to run Cassandra vnodes, otherwise $y_{i,j} = 0$. $x_{i,j,h}$ is the number of Cassandra vnodes serving application i and running on VMs with configuration j allocated on PM h. $s_{i,h}$ is equal to 1 if a Cassandra vnode serving application i run of PM h. Otherwise $s_{i,h} = 0$. Finally, $\mathcal{J} = [1, V] \subset \mathbb{N}$ is the set of VMs configurations indexes, $\mathcal{H} = [1, H] \subset \mathbb{N}$ is the set of PMs indexes and $\mathcal{I} = [1, N] \in \mathbb{N}$ is the set of application indexes.

C. Throughput model

We model the actual throughput T_i offered by the provider to application i as function of $x_{i,j,h}$ (we recall that there is a mapping 1 to 1 between a Cassandra vnode and a VM). From the analysis of data collected from the experiments it emerges that the throughput for a Cassandra vnode serving requests of type l_i and running on a VM of type j (on top of a PM h) can be approximated with a set of linear segment with slope $\delta_{l_i,j}^k$,

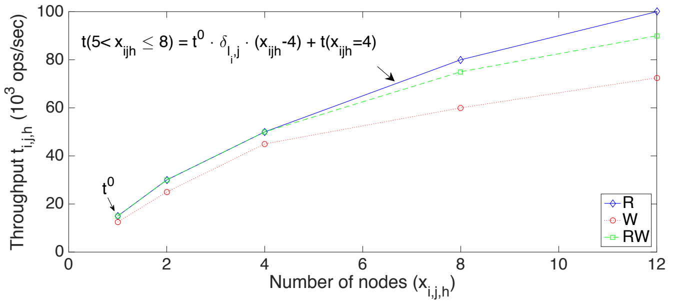

Fig. 1. A real example of Cassandra throughput as function of the number of Cassandra vnodes allocated for different type of requests. The plot shows how the model we propose is realistic.

has shown in Figure 1. $\delta_{l_i,j}^k$ is the slope of the k^{th} segment and is valid for a number of Cassandra nodes between n_{k-1} and n_k. Therefore, for $n_{k-1} \leq x_{i,j,h} \leq n_k$ we can write the following expression:

$$t(x_{i,j,h}) = t(n_{k-1}) + t_{l_i,j}^0 \cdot \delta_{l_i,j} \cdot (x_{i,j,h} - n_{k-1}) \qquad (1)$$

where $k \geq 1$, $n_0 = 1$ and $t_{i,j,h}(1) = t_{l_i,j}^0$. Finally, we define the overall throughput T_i as:

$$T_i(\mathbf{x}) = t\left(n_i\right) \text{ and } n_i = \sum_{j \in \mathcal{J}, h \in \mathcal{H}} x_{i,j,h}, \; \forall i \in \mathcal{I} \qquad (2)$$

where $\mathbf{x} = [x_{i,j,h}]$ $\forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{H}$ and n_i is the number of vnodes used by application i.

D. Power consumption model

As service provider utility we chose the power consumption that is directly related with the provider revenue (and with IT sustainability). In literature has been proposed many work for reducing power and energy consumption in cloud systems, two interesting survey are [16] and [17]. Power consumption models usually define a linear relationship between the amount of power used by a system as function of the CPU utilization (e.g. [18]–[20]), or processor frequency (e.g. [21]) or number of core used (e.g. [22]). In this work we chose a linear model [18] where the power P_h consumed by a physical machine h is function of the CPU utilization and therefore of the system configuration $\mathbf{x}$:

$$P_h(\mathbf{x}) = k_h \cdot P_h^{max} + (1 - k_h) \cdot P_h^{max} \cdot U_h(\mathbf{x}) \qquad (3)$$

where P_h^{max} is the maximum power consumed when the PM h is fully utilised (e.g. 500W), k_h is the fraction of power consumed by the idle PM h (e.g. 70%), and the CPU utilization for PM h is defined by

$$U_h(\mathbf{x}) = \frac{1}{C_h} \cdot \sum_{\mathcal{I}, \mathcal{J}} x_{i,j,h} \cdot c_j \qquad (4)$$

E. The optimization problem

As introduced before the service provider aims to minimize the overall energy consumption $P(\mathbf{x})$ defined by

$$\begin{aligned} P(\mathbf{x}) &= \sum_{h\in\mathcal{H}} P_h(\mathbf{x}) \\ &= \sum_{h\in\mathcal{H}} P_h^{max}\cdot\left(k_h\cdot r_h + \frac{(1-k_h)}{C_h}\cdot\sum_{\mathcal{I},\mathcal{J}} x_{i,j,h}\cdot c_j\right) \end{aligned} \tag{5}$$

where $r_h = 1$ if $x_{i,j,h} > 0$ for some $i \in \mathcal{I}$ and $j \in \mathcal{J}$. Otherwise $r_h = 0$ Therefore, the optimal adaptation plan is described by an instance of $\mathbf{x}$, solution of the following optimization problem

min $f(\mathbf{x}) = P(\mathbf{x})$

subject to:

$$\sum_{\mathcal{J},\mathcal{H}} t(x_{i,j,h}) \geq T_i^{min},\ \forall i \in \mathcal{I} \tag{6}$$

$$\sum_{\mathcal{H}} x_{i,j,h} + \Gamma\cdot(1-y_{i,j}) \geq D_i,\ \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{7}$$

$$x_{i,j,h} \leq \Gamma\cdot y_{i,j},\ \forall i \in \mathcal{I}, j \in \mathcal{J}, h \in \mathcal{H} \tag{8}$$

$$\sum_{\mathcal{J}} y_{i,j} = V,\ \forall i \in \mathcal{I} \tag{9}$$

$$\sum_{\mathcal{I},\mathcal{J}} x_{i,j,h}\cdot c_j \leq C_h,\ \forall\ h \in \mathcal{H} \tag{10}$$

$$\sum_{\mathcal{H}} s_{i,h} \geq D_i,\ \forall i \in \mathcal{I} \tag{11}$$

$$\sum_{\mathcal{J}} x_{i,j,h} - s_{i,h}\cdot\Gamma \leq 0,\ \forall h \in \mathcal{H} \tag{12}$$

$$-\sum_{\mathcal{J}} x_{i,j,h} + s_{i,h} \leq 0,\ \forall h \in \mathcal{H} \tag{13}$$

$$\sum_{\mathcal{I}} s_{i,h} - r_h\cdot\Gamma \leq 0,\ \forall h \in \mathcal{H} \tag{14}$$

$$-\sum_{\mathcal{I}} s_{i,h} + r_h \leq 0,\ \forall h \in \mathcal{H} \tag{15}$$

$$y_{i,j}, s_{i,h} \text{ and } r_h \in [0,1],\ \forall i \in \mathcal{I}, j \in \mathcal{J}, h \in \mathcal{H} \tag{16}$$

$$x_{i,j,k} \in \mathbb{N},\ \forall i \in \mathcal{I}, j \in \mathcal{J}, h \in \mathcal{H} \tag{17}$$

where: Constraints 6 guarantees the SLA is satisfied in term of minimum throughput for all the tenants. For the sake of clarity we keep this constraint non linear, but it can be linearized using standard techniques from operational research if the throughput is modelled using eq. 1. Constraints 7 guarantee that the number of vnodes allocated implement the replication factor specified in the SLA by each tenant. Constraints 8 and 9 model the assumption that for each tenant must be allocated homogeneous VMs and the number of vnodes of the Cassandra datacenter must be greater or equal than D_i. Γ is an extremely large positive number. Constraints 10 control the maximum capacity of the physical machine is not exceeded. A relaxation of such constraint allows to model over-allocation. Constraints 11 guarantee that the Cassandra vnodes are instantiated on at least D_i different physical machines. Constraints 12 and 13 force $s_{i,j}$ to be equal to 1 if the physical machine is used by application i and to be zero on the contrary. In the same way, constraints 14 and 15 force r_h to be equal to 1 if the physical machine is used and zero otherwise. Finally, expressions 16 and 17 are structural constraints of the problem.

III. Sub-optimal adaptation algorithm

In a real scenario it is reasonable that new tenants subscribe the service and/or actual tenants change their SLA (for example requesting the support for an higher throughput or for a different replication factor). In such dynamic scenario the adaptation policy should find the optimal configuration of the Cassandra virtual datacenter without perturbing the performance of the other tenants, that is for example avoiding VMs migration. A limitation of the adaptation model proposed in Sec. II-E is that the re-configuration of a virtual datacenter or the instantiation of a new one can lead to an uncontrolled number of VMs migrations and vertical scaling actions of all the virtual datacenters. Both actions are critical for the performances of the whole datacenter. Two approaches can be used to solve this stability issues. The first one is to embed into the optimization model constraints to avoid or control VMs migration and vertical scaling. An examples to control VM migration is provided in [23], however this model is not linear and was solved with an heuristic that provide a sub optimal solution. Another solution, that always will lead to a sub-optimal solution is to apply locally the optimization problem, that is to optimize the re-configuration/placement only for the interested tenants and only using available resources. Algorithm 1 propose and implementation of the second approach. The proposed heuristic is designed to completely avoid migrations and scaling actions for already allocated virtual datacenters, except the one that eventually demands for SLA variation. The algorithm works on the set $\mathcal{H}_a$ of PMs that have available cores to instantiate Cassandra vnodes. $\mathcal{H}_a$ is updated continuously. The input for the algorithm is the SLA $\mathbf{s} = \langle l_i, T_i^{min}, D_i\rangle$ for a new or a current tenant, and the output produced is the sub-optimal allocation $\mathbf{x}$. At line 3 is evaluated the sub-optimal solution solving the optimization problem for the subset of available resources. If no optimal or sub-optimal solution exist ($e =$ `false`) the request is rejected.

Algorithm 1 Sub-optimal adaptation with zero migrations/reconfigurations

```
Require: $\mathcal{H}_a$; // Set of available nodes in the datacenter
Require: $\{C_{a,j} | j \in \mathcal{H}_a\}$; // Available capacity in the system
1: Input: $\mathbf{s} = \langle l_i, T_i^{min}, D_i\rangle$; // SLA for the new or actual tenant
2: Output: $\mathbf{x}$ // sub-optimal configuration
3: $[\mathbf{x},\ e] \leftarrow$ optAdapt$(C_a, \mathcal{H}_a, \mathbf{s})$
4: if $e$ = false then
5:     $\mathbf{x} \leftarrow \emptyset$ // No feasible solution. The request must be rejected
6: end if
7: return $\mathbf{x}$
```

IV. Performance evaluation

We evaluate the behaviour and the performance of our adaptation model and of the proposed heuristic in three different scenarios:

TABLE I. $t^0_{l_i,j}$ AS FUNCTION OF c_j AND l_i

Type of	VM conf. c_j and related throughput $t^0_{l_i,j}$		
req. (l_i)	8 vcpu	4 vcpu	2 vcpu
R	16.6 $\times 10^3 ops/sec$	8.3 $\times 10^3 ops/sec$	3.3 $\times 10^3 ops/sec$
W	8.3 $\times 10^3 ops/sec$	8.3 $\times 10^3 ops/sec$	3.3 $\times 10^3 ops/sec$
RW	13.3 $\times 10^3 ops/sec$	8.3 $\times 10^3 ops/sec$	3.3 $\times 10^3 ops/sec$

TABLE II. MODEL PARAMETERS

Parameter	Value	Description
N	3 – 12	number of tenants
V	3	number of VM types
H	4 – 16	number of PMs
D_i	2 – 4	replication factor for App. i
$\mathcal{L}$	$\{R, W, RW\}$	set of request types
T_i^{min}	10 – 40 $\times 10^3$ ops/sec	minimum throughput agreed in the SLA
C_h	8	number of cores for PM h
c_j	2 – 8	number of vcores use by VM type j
$[\delta^1_{l_i}, \delta^2_{l_i}, \delta^3_{l_i}]$	[1, 0.8, 0.6] $\forall l_i$	parameter for the model of the throughput $t_{i,j,h}$. In the specific: δ^1 is for up to two nodes; δ^2 is for $x_{i,j,h}$ between 3 and 7; δ^3 is for configurations with 8 vnodes and more.
P_h^{max}	500 Watt	maximum power consumed by PM h if fully loaded
k_h	0.7	fraction of P_h^{max} consumed by PM h if idle

TABLE III. SLA $\langle l_i, T_i^{min}, D_i \rangle$ FOR EACH TENANT i AND THE THROUGHPUT T_i GUARANTEED BY THE SERVICE PROVIDER.

Type of	Light load			Heavy load		
request (l_i)	T_i^{min}	D_i	T_i	T_i^{min}	D_i	T_i
R	10K	4	10.56	18K	4	18.48
W	14K	2	16.6	25.2K	2	26.56
RW	18K	3	18.48	32.4K	3	33.2

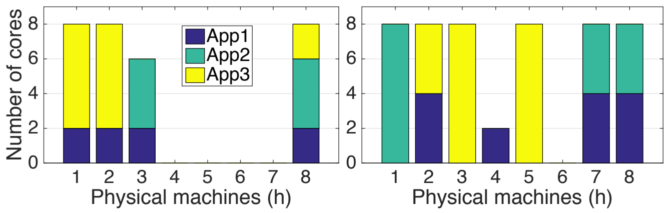

Fig. 2. The resource allocation (in number of cores) for the light and heavy load scenarios.

TABLE IV. AMOUNT AND TYPE OF VMS ALLOCATED FOR EACH APPLICATION.

App.	Light load			Heavy load		
	VMtype	Amount	$Ncores$	VMtype	Amount	$Ncores$
1	3	4	8	3	7	14
2	2	2	8	2	4	16
3	3	7	14	2	5	20

- *Light load and Heavy load case*. This scenario shows in detail why adaptation actions are performed and how them impact the datacenter configuration;
- *New service subscriptions*. This scenario reproduces the arrival of new tenants demanding each for a low intensity throughput
- *Change of SLA*. This scenario reproduces the case of an application that demands for a new SLA with an higher throughput.

We parameterize our model using data measured on a real cluster composed of three physical nodes for a total of 24 cores and 120GB of memory (RAM). We run VMware ESXi 5.5.0 on top of Red Hat Enterprise Linux 6 (64-bit) and we use Cassandra 2.1.5. We use VMs with three different configurations, as reported in Table II. We measure the maximum throughput achievable ($t^0_{l_i,j}$) for each type of workload and VM type (Table I). Moreover, we compute also the values for $\delta_{l_i,j}$ for up to 8 virtual nodes. In the experiments that follow, unless differently specified, we model a datacenter with 8 nodes for a total of 64 cores. The performances of the proposed algorithms are assessed using numerical evaluation. Experiments have been carried on using Matlab R2015b 64-bit for OSX. The model parameters we used in the experiments are reported in Table II. Moreover, we assume that the physical nodes are connected with an high speed LAN and that the workload is not memory bound.

A. Performance metrics

To measure the performance of the adaptation algorithms proposed we consider the following metrics:

- $P(\mathbf{x})$ the overall power consumption defined by equation 5;
- $T_i(\mathbf{x})$ the overall throughput for application i, defined by equation 2. T_i is the actual throughput that can be achieved for application i with a specific datacenter configuration $\mathbf{x}$
- $Ncores_i = \sum_{\mathcal{J},\mathcal{H}} x_{i,j,h} \cdot c_j$ the overall number of virtual cores used by an application i. This metric is used to give a measure of the scaling actions.

B. Light and Heavy load case

The workload submitted to the system is summarized in Table III. We have three different applications with different SLAs. In the heavy load case the throughput requirements are increased by the 80% for each application. How resources are allocated in the two scenarios is reported in Figure 2 and Table IV. The reader can observe the vertical and horizontal scaling actions performed when the volume of requests increase. The allocated resources allow to serve the volume of transactions submitted by the application providing the throughput reported in Table III. Being resources discrete in some cases there is a very limited over-provisioning. The total power consumed is 1962 Watt for the light load case and 3739 for the heavy load case. This example gives to the reader also an idea of how many changes in the system configuration are needed in order to support an heavy workload.

C. New tenants subscription

In this set of experiments we increase the number of tenants starting from 3 up to 7. All the new tenant App demand for the following SLA: $\langle l_i = R, T^m in_i = 10 \times 10^3 ops/sec, D_i = 3 \rangle$. Figure 3 compares the Cassandra clusters configuration achieved using the optimal allocation policy (top row) and the configuration achieved using the proposed heuristic (bottom row). As

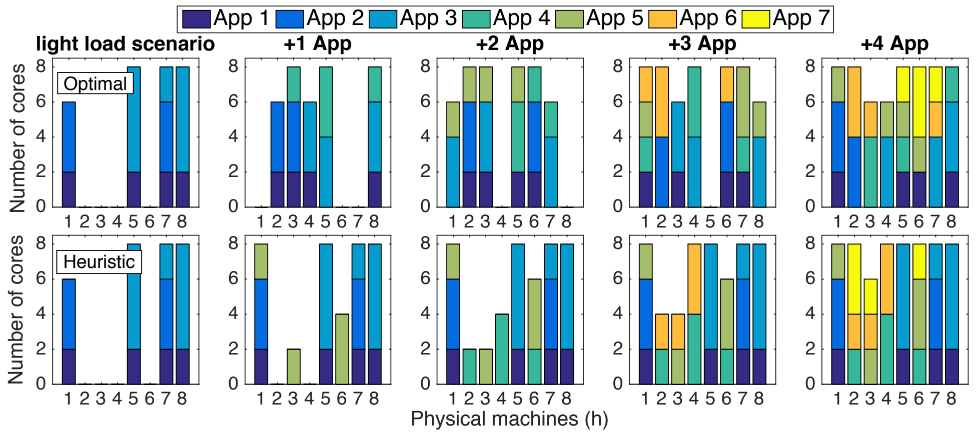

Fig. 3. Comparison between the allocation achieved with the optimal policy (top row) and the allocation achieved with the heuristic (bottom row)

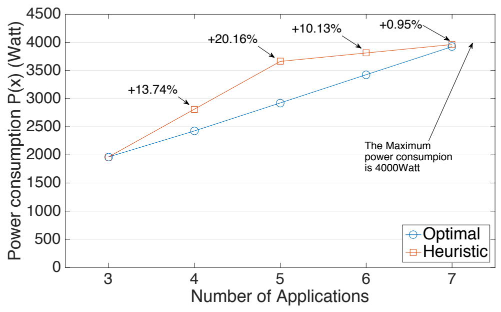

Fig. 4. Total Power consumption for Optimal adaptation and Heuristic adaptation

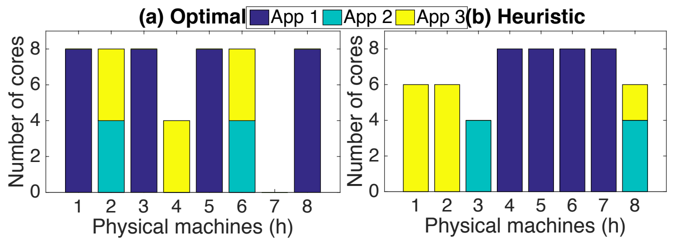

Fig. 5. Optimal and sub-optimal allocation in case of 50000 ops/sec.

expected, the optimal allocation generate a new configuration of the Cassandra clusters at each adaptation step that results in a new placement of the Cassandra nodes. On the contrary, the heuristic does not change the previous placement. We can observe that when we add a new tenant, moving from the *light load scenario* to the *+1 App scenario* the optimal allocation policy did a vertical adaptation action moving from 7 VMs with configuration type 3 (14 cores) to 3 VMs with configuration type 2 (12 cores), saving 2 cores. This justify the different allocation obtained in scenario *+4 App*. In term of power consumption the gap between optimal and sub-optima adaptation is reported in figure 4 and range between 10% and 20%. This case shows that must be chosen a trade off between what we can gain from an optimal configuration and the disruptive action we should take to implement it.

D. Change of SLA

In this set of experiments we change the SLA of application 1 increasing the throughput demand from 10×10^3 ops/sec to 60×10^3 ops/sec, and we compare the performance of the optimal adaptation policy and of the proposed heuristic in managing SLA changes. Applications 2 and 3 has always the same SLA specified in the light load scenario. Concerning the optimal adaptation: for application 2 no adaptation action at all are taken; for application 3 there is a vertical scaling action (from 7 VMs of type 3 to 3 VMs of type 2) that reduce the number of core used. For application 1, there is immediately a change in node configuration (from 3 to 2) and for 50000 ops/sec and up the vnodes configuration is changed again and VM of type 1 is used. When we run the heuristic, for application 2 and 3 no adaptation is performed and for application 1 the adaptation actions are the same that in the optimal case. However, because for application 3 are ever used 7 VMs of type 3 there is no available capacity to allocate enough resources when the volume of requests is 60000ops/sec. Figure 5 show the optimal and sub-optimal configuration obtained for a volume of requests equal to 50000ops/sec. It is evident that the case of sub-optimal allocation there is no free space for one more VM of type 1 that is needed to serve a volume of requests equal to 60000ops/sec. Finally, Figure 6 shows the power consumed by the system for the different volumes of requests. The cost payed for not perturbing the system each time a new Cassandra cluster is instantiate is around the 20% higher than the optimal case. Higher the ratio between the CPU cores demanded by the application and the available cores, and lower the penalty in term of power consumption.

V. Concluding remarks

In this paper we explore the problem of autonomic energy-aware adaptation of multi-tenant Cassandra based systems. We proposed an optimization model and a heuristic to find a sub-optimal solution with the goal of avoiding system perturbations, possible cause of performance degradation. The main advantages of the model we propose is that it need only to know the relationship between the throughput and the number of Cassandra vnodes. This information is easy to be collect and maintained up to date at execution time. Our

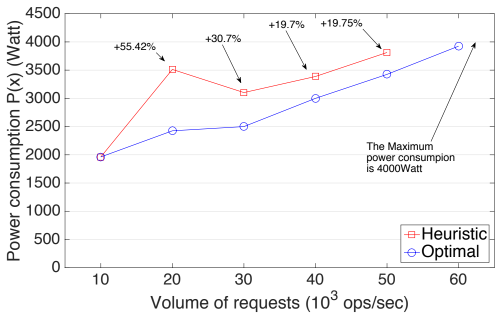

Fig. 6. Power consumption $P(\mathbf{x})$ for increasing volume of requests from Application 1.

main finding are the following. First, the proposed model has a correct behaviour in different dynamic scenarios. As expected, it's drawback is the entropy that successive adaptation actions can create. Second, the heuristic we propose is capable to find a feasible solution with a penalty, in term of power consumption, between 10 and 30%. The penalty depend on the ratio between the amount of available resources and the application's demand, and we guess it can be reduced enabling the reconfiguration of a small subset of tenants.

Acknowledgments

This work is funded by the research project "Scalable resource-efficient systems for big data analytics" funded by the Knowledge Foundation (grant: 20140032) in Sweden Moreover, the authors would like to thanks Jim Hakansson and Ericsson AB, Sweden for their support.

References

[1] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 143–154. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807152

[2] S. Patil, M. Polte, K. Ren, W. Tantisiriroj, L. Xiao, J. López, G. Gibson, A. Fuchs, and B. Rinaldi, "Ycsb++: Benchmarking and performance debugging advanced features in scalable table stores," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 9:1–9:14. [Online]. Available: http://doi.acm.org/10.1145/2038916.2038925

[3] T. Rabl, S. Gómez-Villamor, M. Sadoghi, V. Muntés-Mulero, H.-A. Jacobsen, and S. Mankovskii, "Solving big data challenges for enterprise application performance management," *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 1724–1735, Aug. 2012. [Online]. Available: http://dx.doi.org/10.14778/2367502.2367512

[4] Y. Shi, X. Meng, J. Zhao, X. Hu, B. Liu, and H. Wang, "Benchmarking cloud-based data management systems," in *Proceedings of the Second International Workshop on Cloud Data Management*, ser. CloudDB '10. New York, NY, USA: ACM, 2010, pp. 47–54. [Online]. Available: http://doi.acm.org/10.1145/1871929.1871938

[5] J. Kuhlenkamp, M. Klems, and O. Röss, "Benchmarking scalability and elasticity of distributed database systems," *Proc. VLDB Endow.*, vol. 7, no. 12, pp. 1219–1230, Aug. 2014. [Online]. Available: http://dx.doi.org/10.14778/2732977.2732995

[6] E. Dede, M. Govindaraju, D. Gunter, R. S. Canon, and L. Ramakrishnan, "Performance evaluation of a mongodb and hadoop platform for scientific data analysis," in *Proceedings of the 4th ACM Workshop on Scientific Cloud Computing*, ser. Science Cloud '13. New York, NY, USA: ACM, 2013, pp. 13–20. [Online]. Available: http://doi.acm.org/10.1145/2465848.2465849

[7] M. Chalkiadaki and K. Magoutis, "Managing service performance in nosql distributed storage systems," in *Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing*, ser. MW4NG '12. New York, NY, USA: ACM, 2012, pp. 5:1–5:6. [Online]. Available: http://doi.acm.org/10.1145/2405178.2405183

[8] P. Shankaranarayanan, A. Sivakumar, S. Rao, and M. Tawarmalani, "Performance sensitive replication in geo-distributed cloud datastores," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, June 2014, pp. 240–251.

[9] J. a. Paiva, P. Ruivo, P. Romano, and L. Rodrigues, "Autoplacer: Scalable self-tuning data placement in distributed key-value stores," *ACM Trans. Auton. Adapt. Syst.*, vol. 9, no. 4, pp. 19:1–19:30, Dec. 2014. [Online]. Available: http://doi.acm.org/10.1145/2641573

[10] N. Diegues, M. Orazov, J. a. Paiva, L. Rodrigues, and P. Romano, "Optimizing hyperspace hashing via analytical modelling and adaptation," *SIGAPP Appl. Comput. Rev.*, vol. 14, no. 2, pp. 23–35, Jun. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656864.2656866

[11] F. Almeida Morais, F. Vilar Brasileiro, R. Vigolvino Lopes, R. Araujo Santos, W. Satterfield, and L. Rosa, "Autoflex: Service agnostic auto-scaling framework for iaas deployment models," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, 2013, pp. 42–49.

[12] Y. Jiang, C.-S. Perng, T. Li, and R. N. Chang, "Cloud analytics for capacity planning and instant vm provisioning," *Network and Service Management, IEEE Transactions on*, vol. 10, no. 3, pp. 312–325, 2013.

[13] H. Ghanbari, B. Simmons, M. Litoiu, C. Barna, and G. Iszlai, "Optimal autoscaling in a iaas cloud," in *Proceedings of the 9th international conference on Autonomic computing*, ser. ICAC '12. New York, NY, USA: ACM, 2012, pp. 173–178. [Online]. Available: http://doi.acm.org/10.1145/2371536.2371567

[14] E. Casalicchio and L. Silvestri, "Mechanisms for sla provisioning in cloud-based service providers," *Computer Networks*, vol. 57, no. 3, pp. 795 – 810, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128612003763

[15] N. Grozev and R. Buyya, "Multi-cloud provisioning and load distribution for three-tier applications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 9, no. 3, p. 13, 2014.

[16] B. Priya, E. Pilli, and R. Joshi, "A survey on energy and power consumption models for greener cloud," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, Feb 2013, pp. 76–82.

[17] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos, "Cloud computing: Survey on energy efficiency," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 33:1–33:36, Dec. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656204

[18] R. Buyya, A. Beloglazov, and J. H. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *CoRR*, vol. abs/1006.0308, 2010. [Online]. Available: http://arxiv.org/abs/1006.0308

[19] D. Borgetto, M. Maurer, G. Da-Costa, J. Pierson, and I. Brandic, "Energy-efficient and sla-aware management of iaas clouds," in *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, May 2012, pp. 1–10.

[20] A. Dalvandi, M. Gurusamy, and K. C. Chua, "Time-aware vmflow placement, routing, and migration for power efficiency in data centers," *Network and Service Management, IEEE Transactions on*, vol. 12, no. 3, pp. 349–362, Sept 2015.

[21] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. Khan, "Greencloud: A packet-level simulator of energy-aware cloud computing data centers," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec 2010, pp. 1–5.

[22] L. A. Rocha and E. Cardozo, "A hybrid optimization model for green cloud computing," in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, ser. UCC '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 11–20. [Online]. Available: http://dx.doi.org/10.1109/UCC.2014.9

[23] E. Casalicchio, D. A. Menascé, and A. Aldhalaan, "Autonomic resource provisioning in cloud systems with availability goals," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, ser. CAC '13. New York, NY, USA: ACM, 2013, pp. 1:1–1:10. [Online]. Available: http://doi.acm.org/10.1145/2494621.2494623